PharmaSUG 2024 - Paper AP-268

A New Approach to Automating the Creation of the Subject Visits (SV) Domain

Xiangchen (Bob) Cui, Jessie Wang, and Min Chen, CRISPR Therapeutics AG, Boston, MA

ABSTRACT

The creation of the subject visits (SV) domain is one of the most challenging tasks of SDTM programming. Aside from the small portion of mapping from raw dataset variables to SV variables, SV programming mainly consists of a more complex derivation process, which is totally different from that of other SDTM domains. The dynamic parts of the SV programming process, such as identifying raw datasets and their variables with both date/time and clinical visits, cause manual development of a SAS program to be time-consuming and error prone. Hence, automating its code generation would achieve and enhance efficiency and accuracy.

This paper will present a new approach for SV automation based on the SDTM automation done in our previous paper, which leveraged CRF specifications from an EDC database and SDTM standards [1]. It will introduce the standard SV programming logic flow with 10 sequential steps, which leads us to develop an additional SAS-based macro named ***%SV_Code_Generator*** as an expansion to the macro introduced in [1]. The output of this macro (SV.sas) achieves 100% automation of SV domain for the raw data collected per CRFs in a clinical study.

This new approach guarantees all raw dataset variables related to subject visits are accounted for in SV programming thanks to the sequential programming automations. This automation allows for the generation of SV dataset to occur very early in the programming development cycle and makes developing programmatic quality checks for clinical data review and data cleaning more efficient and economically feasible.

INTRODUCTION

A data-driven approach utilizing SASHELP views and CALL EXECUTE statements to bring in dates programmatically and dynamically while building the SV domain is believed to be more robust than manual identification and typical data step programming [2]. However, it still is a manual process, instead of an automation through a metadata-driven method.

In our previous paper, we introduced a new approach in automatic SAS code generation to achieve SDTM automation by a metadata-driven method leveraging both CRF specifications from an EDC database and SDTM standards [1]. The macro named ***%SDTM_Code_Generator*** generates a SDTM mapping SAS program after each call. However, since SV is a special purpose domain that requires complex derivations, more work is warranted to automate SV programming.

This paper will present our thought process in developing ***%SV_Code_Generator*** and provide details on the standard SV programming logic flow with 10 sequential steps and the rationale of the automatic generation of SV domain. A deep understanding of SV programming has led us to develop two steps for the automatic SV programming process as follows:

1. Call ***%SDTM_Code_Generator*** to generate SAS code for mapping visit-related raw data to the SV domain.
2. Call ***%SV_Code_Generator*** to generate SAS code for complex derivations and combine it with the SAS code from Step 1 to automatically output a SAS program for the SV domain (SV.sas).

This paper will illustrate how the two macros work together to implement the 10 sequential steps of the SV programming process. It will identify which parts of the process are “dynamic” and which parts are “standard” and provide snippets of SAS code to demonstrate how to process these parts. It will also show how all variables collected for the SV domain are accounted for in SV programming through the sequential programming automations built into the programming process. Finally, it will introduce how to

handle external data and how to validate the SV dataset generated by ***%SV_Code_Generator*** to achieve both high quality and efficiency.

Samples of ***_SV.sas*** generated from ***%SDTM_Code_Generator*** and ***SV.sas*** generated from %***SV_Code_Generator*** are included in Appendix 2 and 3, respectively, for the illustration of the new approach throughout this paper and for easy reference.

IMPORTANCE OF THE SV DOMAIN'S COMPLIANCE WITH CDISC STANDARDS AND FDA VALIDATOR RULES

The CDISC SDTM Implementation Guide Version 3.4 (SDTMIG v3.4) states that Subject Visits (SV) is *"[a] special purpose domain that contains information for each subject's actual and planned visits. The Subject Visits domain consolidates information about the timing of subject visits that is otherwise spread over domains that include the visit variables […] Unless the beginning and end of each visit is collected, populating the SV dataset will involve derivations. In a simple case, where, for each subject visit, exactly 1 date appears in every such domain, the SV dataset can be created easily by populating both SVSTDTC and SVENDTC with the single date for a visit. When there are multiple dates and/or date/times for a visit for a particular subject, the derivation of values for SVSTDTC and SVENDTC* ***may be more complex****."* [3]

The FDA has published FDA Validator Rules and periodically updated them since March 2017. For the SV domain, Validator Rule **SD0065** states that "*[all] Unique Subject Identifier (USUBJID) + Visit Name (VISIT) + Visit Number (VISITNUM) combination values in data should be present in the Subject Visits (SV) domain*" [4]. The FDA uses the Study Data Technical Conformance Guide to express the expectation that sponsors should comply with these rules (as well as the SDTMIG) when submitting SDTM datasets for regulatory review and analysis [5].

Thus, it is crucial for sponsors to carefully identify which dates should be included in derivations for SVSTDTC and SVENDTC, determine the logic flow for those derivations, and ensure that all visits are properly accounted for in the SV domain. This can be a tedious process as those numerous dates and visits come from multiple sources, and accidental omissions of data can occur. There are two different approaches to programming SV to comply with these rules.

TWO APPROACHES TO PROGRAMMING THE SV DOMAIN

The two approaches to programming SV are: 1) generate SV once all other SDTM domains with visit information are available as inputs and 2) generate SV with raw data as inputs. Table 1 summarizes the pros and cons of these two approaches.

Approach	The Input of Programming /Derivation	Pros	Cons
Approach 1	SDTM Datasets	Straightforward and simple for deriving SV	Circularity: Must go back to each SDTM domain with visits to update VISIT and VISITNUM for unscheduled visits to comply with FDA Validator Rule: **SD0065**
Approach 2	Raw Datasets	Avoiding circularity in the entire SDTM programming and easily complying with FDA Validator Rule: **SD0065**	Prone to human errors and time consuming in identifying raw datasets and their variables to be used

Table 1. The Summary of Pros and Cons of Two Approaches to Programming SV Domain

[6] strongly recommends that SV should be populated first from raw data to avoid circularity in SDTM programming and to comply with FDA Validator Rule: SD0065. We agree with that recommendation and have chosen approach 2 for our SV programming process.

INTRODUCTION TO STANDARD SV PROGRAMMING LOGIC FLOW

Before starting to program the SV domain, one should first understand the general SV programming logic flow before trying to standardize and further automate the process.

Display 1 below shows 10 sequential steps followed in SAS SV programming, and Figure 1 below depicts the overall logic and data flow of these 10 steps.

1. **Identify raw datasets with both dates/times and clinical visits.**
2. **Separate them into two data blocks: one with dates only, another with both dates and times.**
3. **For each data block, stack the raw datasets and standardize their date variable names into a common name (SVDTC). If needed, do preprocessing for variables used in Step 9.**
4. Combine these two data blocks to derive VISIT and VISITNUM for scheduled visits and derive SV variables describing visit-related information: SVPRESP, SVOCCUR, SVREASOC per SDTMIG v3.4.

   4.5 Import external datasets (if available) and combine them with the above two data blocks.
5. Separate the combined data into two data blocks: one with scheduled visits only, another with unscheduled visits only.
6. For each scheduled visit, derive SVSTDTC from the earliest date/time and derive SVENDTC from the latest date/time. For each unscheduled visit, set both SVSTDTC and SVENDTC equal to SVDTC.
7. Combine the scheduled visit and unscheduled visit blocks and derive VISIT and VISITNUM for unscheduled visits.
8. Derive SVSTDY and SVENDY.
9. **Map other raw dataset variables into SUPPSV.**
10. Output permanent datasets: SV and SUPPSV.

Display 1. 10 Sequential Steps in SAS SV programming.

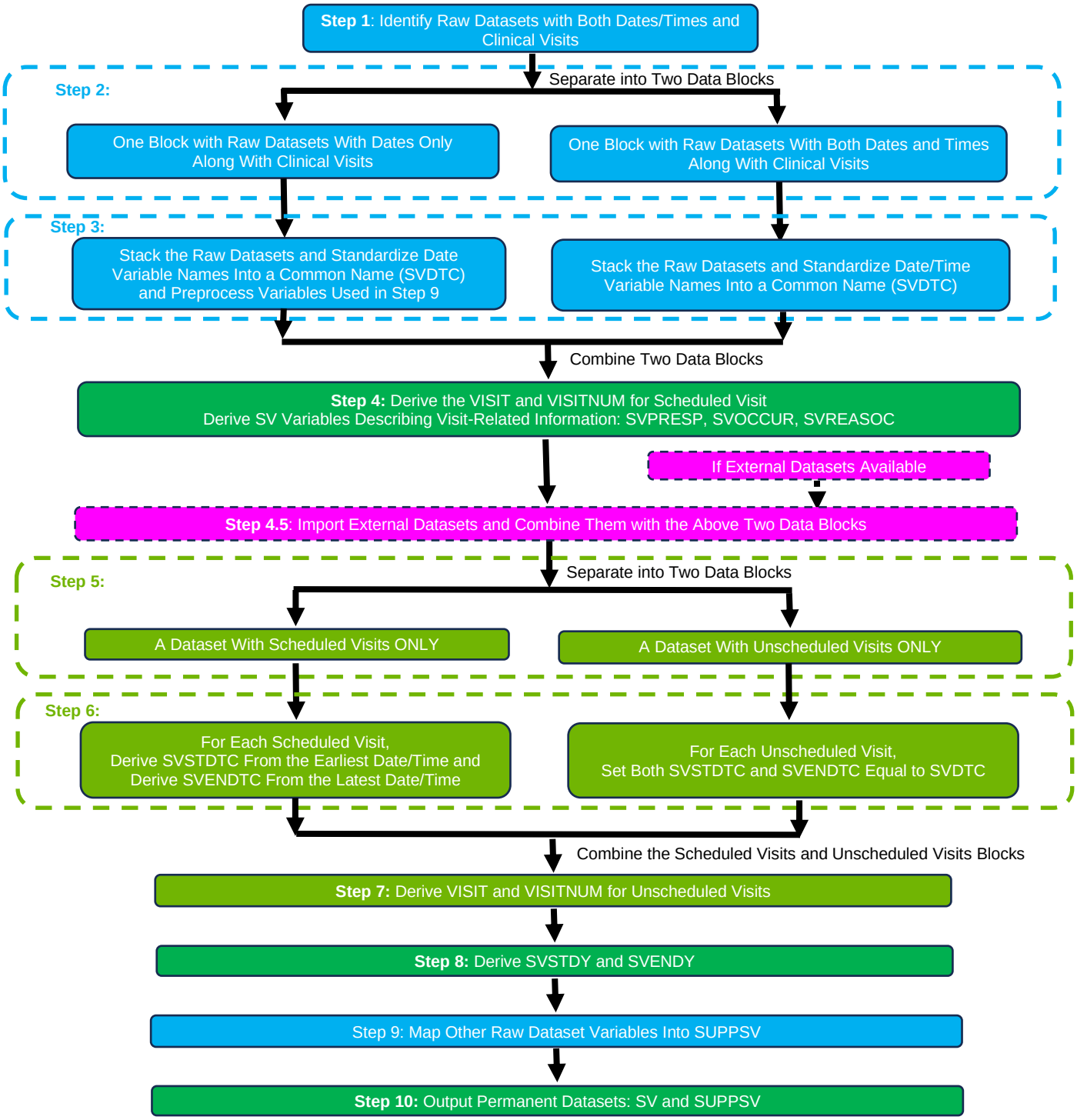


Figure 1. General SV Programming Logic Flow

RATIONALE FOR THE AUTOMATIC CREATION OF THE SV DOMAIN

The advantages and benefits of a macro for SDTM automation over SDTM mapping template SAS programs have been fully explained and demonstrated in [1]. Furthermore, from Figure 1 above, one can easily identify that Steps 4, 5, 6, 7, 8, and 10 are "***standard***" SV programming, which means that they do not change on a study-by-study basis. However, Steps 1, 2, 3, and 9 are "**dynamic**"; they're study-

dependent because different studies could have different CRF/EDC designs and raw dataset variable names. Manually developing SAS code for Steps 1, 2, 3, and 9 is a time-consuming and labor-intensive process, which is prone to human errors. If the generation of SAS code for Steps 1, 2, 3, and 9 could be automated, the accuracy and efficiency of SV domain programming could be improved and enhanced. The following example illustrates it!

In one of our studies, there are 80 raw datasets with over 900 variables. Of those datasets, 75 datasets contain a total of 143 date/time variables. Among those datasets, 66 are associated with clinical visits: 57 with only dates and 9 with both dates and times. The "select" statement in Display 2 below shows an example of the 66 CRFs/raw datasets with both dates/times and visits along with a 67th dataset (NV), which does not contain any dates but does contain visit-related data that should be mapped to a supplemental qualifier. These datasets are the inputs for SV programming. Please refer to the block "Step 1: Identify Raw Datasets with Both Date/Time and Clinical Visits" in Appendix 3 to see the full context of this code.

```
proc datasets lib=work kill nolist nowarn;
     copy in=raw out=work;
          select ANC BIO BIONON BS CLLSLL CONS CP CY DIA ECHO ECOG EG EQ5D5L EX1 EX2 EX3
                 FACT FACTLEU ICE INL INTL1 INTL2 ITL1 ITL2 IWCLL LBBC LBBMCLL LBBMLYM LBCHEM LBCOAG LBCRS LBHM LBIMMU
                 LBLYM LBPREG LBSER LP LS LUGANO MR MRD MRI NL NTL1 NTL2 NV ORG PE PET1 PET2
                 PK PKINF PROG RADPOST RADPRE RETX SCTPOST SCTPRE SS SV TB THERPOST THERPRE TL1 TL2 UV VS;
     copy in=sdtm out=work;
          select dm;
quit;
```

Display 2. SAS Code from SV.sas Selecting 67 Raw Datasets to be Included in SV

Manually identifying and including all these forms in SV programming for every study is inefficient and prone to human errors. Furthermore, any EDC database changes to these raw dataset names and their date-related variables would require corresponding updates within SAS programs. A macro, on the other hand, could automatically account for these changes. Information on how to generate this SAS code will be provided in the later sections detailing the automation for Steps 1, 2, 3, and 9.

Furthermore, automation allows for SV to be generated early in the programming development cycle. In contrast, manual SV generation would typically occur later in the development cycle (e.g., closer to database lock or before FDA submission) as it's better to wait for more complete data before manually working on SV SAS programming. This earlier automatically generated SV dataset can then be used to develop programmatic quality checks to support clinical data review and data cleaning efforts.

INTRODUCTION TO *%SDTM_CODE_GENERATOR* AND ITS FUNCTION IN SV PROGRAMMING AUTOMATION

[1] introduced our ***%SDTM_Code_Generator*** macro, which uses the master-annotation spreadsheet and SDTM domain specifications as inputs to automatically generate SAS code for SDTM datasets. The generated SAS code is saved to both output datasets and SAS program files. Please refer to [1] for the rationale behind developing the macro, the new SDTM programming workflow, its programming validation process, the master-annotation, and the scalability of this SDTM automation, etc.

For the SV domain, ***%SDTM_Code_Generator*** can be used to automatically generate code to map raw datasets that specifically collect visit-related data. However, it does not have the functionality for automatically deriving SVSTDTC, SVENDTC, VISITNUM, and VISIT. Thus, we created an additional macro named ***%SV_Code_Generator*** to handle the additional derivations. The rationale for creating a separate macro is quoted as follows:

"The SUBJECT VISITS (SV) domain is a special purpose domain that requires more complex derivations, many of which are different from ones of macro ***%SDTM_Code_Generator****. To simplify the development of the macro and reduce the length of SAS code needed, we developed an additional macro named* ***%SV_Code_Generator****, which leverages the output from the call of* ***%SDTM_Code_Generator*** *and extends it further"* [1].

Appendix 2 shows an example of the SAS program _SV.sas which is generated from the calling of ***%SDTM_Code_Generator(domain_=SV)***. Certain sections of _SV.sas (labeled as “Block 1”, “Block 2”, “Block 3”, “Block 4”, and “Block 5”) are used by the new macro ***%SV_Code_Generator*** to generate and output SV.sas, which is shown in Appendix 3. Annotations in SV.sas (Appendix 3) indicate which sections came from the various blocks in _SV.sas (Appendix 2). SV.sas (Appendix 3) is then used to generate the SV domain dataset.

The following sections will introduce the master-annotation (in particular, the section dedicated to the SV domain), the SV domain specification, and the new macro ***%SV_Code_Generator***.

INTRODUCTION TO OUR MASTER-ANNOTATION SPREADSHEET AS RELATED TO THE SV DOMAIN

As discussed in our other paper, the master-annotation spreadsheet is one of the inputs for ***%SDTM_Code_Generator*** [1]. With CRF specifications as its foundation, this spreadsheet contains all raw dataset names, variable attributes (variable names, labels, types, etc.), and annotations for mapping these raw dataset variables to specific SDTM domains. We also added additional columns to the spreadsheet to aid ***%SDTM_Code_Generator*** in automating SAS code generation. Please see Table 2 below for key columns of the master-annotation and Table 3 for a portion of key variables in the master-annotation for the SV domain. Appendix 1 contains the corresponding annotated CRFs for these visit-related forms: Subject Visit (SV), Unscheduled Subject Visit (UV), and Next Visit (NV). Raw dataset variable names are annotated in blue text while SDTM mappings are annotated in red text.

From CRF Specifications, e.g., ALS | **From CRF Annotation** | **Assisting the Macro to Automate the SAS Code Generation**

EDC DATASET NAME	EDC DATASET LABEL	ORD.	VARIABLE TYPE	VARIABLE NAME	VARIABLE LABEL	SDTM DOMAIN	SDTM VARIABLE	QLABEL	QORIG	TRT ASSIGN
NV	Next Visit	1	char	NVYN_STD	Subject Be Advancing to the Next Visit	SV	NXTVISYN in SUPPSV	Subject Advancing to Next Visit?	CRF	
SV	Subject Visit	1	char	VISYN_STD	Was visit performed?	SV	[NOT SUBMITTED]			
SV	Subject Visit	2	Date	VISDAT	Date of Visit	SV	VISDTC in SUPPSV	Visit Date	CRF	
SV	Subject Visit	3	char	VISREASOC_STD	Reason Visit not Performed	SV	SVREASOC			
SV	Subject Visit	4	char	VISREASOC_AE	Adverse Event, Specify	SV	VISMAEID in SUPPSV	Missed Visit: Adverse Event, Specify	CRF	
SV	Subject Visit	5	char	VISREASOC_O	Other Reason, Specify	SV	VISMOTHS in SUPPSV	Missed Visit: Other Reason, Specify	CRF	
SV	Subject Visit	6			Type of Visit check all that apply	SV				
SV	Subject Visit	7	Numeric	VISCNTMDINC	In-Clinic	SV	SVCNTMOD			
SV	Subject Visit	8	Numeric	VISCNTMDINH	In-Home	SV	SVCNTMOD			
SV	Subject Visit	9	Numeric	VISCNTMDR	Remote	SV	SVCNTMOD			
SV	Subject Visit	10	Numeric	VISCNTMDOT	Other	SV	SVCNTMOD			
SV	Subject Visit	11	char	VISCNTMD_O	Type of Visit, Other	SV	VISTYPOS in SUPPSV	Type of Visit: Other, Specify	CRF	
SV	Subject Visit	12	char	VISEPCHGI_STD	Epidemic/Pandemic Related Change	SV	SVEPCHGI			
UV	Unscheduled Subject Visit	1	Date	UVDAT	Date of Visit	SV	SVSTDTC			
UV	Unscheduled Subject Visit	2	char	UVREAS_STD	Reason for Unscheduled Visit	SV	SVUPDES			
UV	Unscheduled Subject Visit	3	char	UVREAS_O	Other, Specify	SV	UNSREASO in SUPPSV	Unscheduled Visit: Other Reason, Specify	CRF	
UV	Unscheduled Subject Visit	4			Unscheduled Assessments Performed	SV				
UV	Unscheduled Subject Visit	7	Numeric	BS	Binet Staging	SV	UNSAPERF in SUPPSV	Unscheduled Assessments Performed	CRF	COMBINE
UV	Unscheduled Subject Visit	10	Numeric	MRI	Brain MRI	SV	UNSAPERF in SUPPSV	Unscheduled Assessments Performed	CRF	COMBINE
UV	Unscheduled Subject Visit	14	Numeric	CY	Cytokines	SV	UNSAPERF in SUPPSV	Unscheduled Assessments Performed	CRF	COMBINE
UV	Unscheduled Subject Visit	15	Numeric	EG	ECG	SV	UNSAPERF in SUPPSV	Unscheduled Assessments Performed	CRF	COMBINE
UV	Unscheduled Subject Visit	17	Numeric	EQ5D5L	EQ-5D-5L	SV	UNSAPERF in SUPPSV	Unscheduled Assessments Performed	CRF	COMBINE
UV	Unscheduled Subject Visit	25	Numeric	NL	Lesion Assessment - New Lesion	SV	UNSAPERF in SUPPSV	Unscheduled Assessments Performed	CRF	COMBINE
UV	Unscheduled Subject Visit	27	Numeric	TL1	Lesion Assessment - Target Lesions - Baseline	SV	UNSAPERF in SUPPSV	Unscheduled Assessments Performed	CRF	COMBINE
UV	Unscheduled Subject Visit	29	Numeric	TL2	Lesion Assessment - Target Lesions - Post-Baseline	SV	UNSAPERF in SUPPSV	Unscheduled Assessments Performed	CRF	COMBINE

EDC DATASET NAME	EDC DATASET LABEL	ORD.	VARIABLE TYPE	VARIABLE NAME	VARIABLE LABEL	SDTM DOMAIN	SDTM VARIABLE	QLABEL	QORIG	TRT ASSIGN
UV	Unscheduled Subject Visit	32	Numeric	CHEM	Local Lab Chemistry	SV	UNSAPERF in SUPPSV	Unscheduled Assessments Performed	CRF	COMBINE
UV	Unscheduled Subject Visit	35	Numeric	HEM	Local Lab Hematology	SV	UNSAPERF in SUPPSV	Unscheduled Assessments Performed	CRF	COMBINE
UV	Unscheduled Subject Visit	47	Numeric	IWCLL	Response Assessment iwCLL	SV	UNSAPERF in SUPPSV	Unscheduled Assessments Performed	CRF	COMBINE
UV	Unscheduled Subject Visit	48	Numeric	LUGANO	Response Assessment Lugano	SV	UNSAPERF in SUPPSV	Unscheduled Assessments Performed	CRF	COMBINE
UV	Unscheduled Subject Visit	50	Numeric	TB	Tumor Biopsy /Cytology/ Pathology	SV	UNSAPERF in SUPPSV	Unscheduled Assessments Performed	CRF	COMBINE
UV	Unscheduled Subject Visit	51	Numeric	VS	Vital Signs	SV	UNSAPERF in SUPPSV	Unscheduled Assessments Performed	CRF	COMBINE

Table 2. Key Columns of the Master-Annotation Dedicated to the SV Domain (With Raw Datasets: NV, SV, and UV)

Column	Column Content		Origin	Manual?
EDC DATASET NAME	Raw Dataset Name		ALS	
EDC DATASET LABEL	Raw Dataset Label		ALS	
ORDER	The Order of Variables specified in CRFs, and One of Keys to Sort Intermediate Datasets for Writing SAS Programs from the Macro Calls		ALS	
VARIABLE TYPE	Variable Type in Raw Dataset, Numeric, char, Date, Time, or Date & Time		ALS	Derived
VARIABLE NAME	Variable Name in Raw Dataset		ALS	
VARIABLE LABEL	Variable Label in Raw Dataset		ALS	
SDTM VARIABLE	SDTM Variable Name, SDTM Variable Name for A Specific Test, QNAM in Supplemental Domain, or Not Submitted		CRF Annotation	Y
QLABEL	Assign QLABEL in Supplemental Domains	Triplet to Help Mapping Raw Dataset Variables in Supplemental Domains	Assisting Macro to Generate SAS codes for SDTM generation	Y
QORIG	Assign QORIG in Supplemental Domains, with Values: CRF, Derived, or Assigned.			
QEVAL	Assign QEVAL in Supplemental Domains, e.g., 'CLINICAL STUDY SPONSOR'			
TRT ASSIGN	Column to aid automation, indicating extra coding is needed for the mapping of the variables, Applicable to all finding domains, and DS			

Table 3. A Portion of the Key Variables in the Master-Annotation Dedicated to the SV Domain [1]

Of note, the column TRT ASSIGN is used to indicate that additional coding is needed for a specific variable. In particular, the SV domain uses TRT ASSIGN = "COMBINE" to concatenate the values of multiple raw dataset variables before mapping them into a supplemental qualifier with QNAM = "UNSAPERF". The following section titled "*Automation Step 9: Map Other Raw Dataset Variables Into SUPPSV*" provides a detailed explanation of the rationale and how it is handled by the new macro.

INTRODUCTION TO OUR STANDARD SDTM SV DOMAIN SPECIFICATION

[1] also introduced our standard format for SDTM specifications [7], which is based on CDISC standards. See Table 4 below for an example of the SV domain specification. Its first six columns: Variable, Label, Type, Controlled Terminology, and Core come directly from the SDTMIG v3.4. Per SDTMIG [3], the sources of SDTM variables are categorized by the origin of the data source in the Define-XML document file as "CRF", "Protocol", "Assigned", or "Derived". The last column Derivation/Assigned is added to facilitate Define-XML generation and customize our code for SDTM automation. When one line of customization code is sufficient, ***%SDTM_Code_Generator*** directly writes that line of code to _SV.sas (e.g., SVPRESP and SVCNTMOD). However, when variables require more complicated code (e.g., EPOCH and SVSTDY), ***%SDTM_Code_Generator*** retrieves the macro calls and writes a line of code to _SV.sas for each macro call (e.g., ***%get_epoch*** and ***%get_dy***). Please refer to [1] for the detailed introduction to SAS utility macros dedicated to the automation of SDTM programs. Display 3 shows the generated macro calls for _SV.sas along with the programming comments. This code can also be found in Block 3 of Appendix 2, which contains the full _SV.sas.

```
*** Programming Note: SDTM Variable: SVSTDY Needs the Derivation by the Macro Call: %get_dy;
%get_dy(_DATEVAR=SVSTDTC, _DAYVAR=SVSTDY);
*** Programming Note: SDTM Variable: SVENDY Needs the Derivation by the Macro Call: %get_dy;
%get_dy(_DATEVAR=SVENDTC, _DAYVAR=SVENDY);
```

Display 3. SAS Code Generated by *%SDTM_Code_Generator* for the Macro Calls of the SV Domain

Variable	Label	Type	Controlled Terminology	Origin	Core	Derivation/Assigned
STUDYID	Study Identifier	Char		Protocol	Req	STUDYID = 'Project-Study-101';
DOMAIN	Domain Abbreviation	Char	DOMAIN	Assigned	Req	DOMAIN = 'SV';
USUBJID	Unique Subject Identifier	Char		Derived	Exp	USUBJID = strip(STUDYID) ||strip(substr(SUBJECT,4));
VISITNUM	Visit Number	Num		Assigned	Req	visitnum = input(visit, ??visitnum.);
VISIT	Visit Name	Char		Assigned	Perm	visit = strip(put(folder, $visit.));
SVPRESP	Pre-specified	Char	NY	Assigned	Exp	if VISIT ^= 'Unscheduled' then SVPRESP = 'Y';
SVOCCUR	Occurrence	Char	NY	Assigned	Exp	
SVREASOC	Reason for Occur Value	Char		CRF Page	Perm	
SVCNTMOD	Contact Mode	Char	CNTMODE	Derived	Perm	SVCNTMOD = strip(coalescec(put(VISCNTMDINC, CNTMDINC.), put(VISCNTMDINH, CNTMDINH.), put(VISCNTMDR, CNTMDR.), put(VISCNTMDOT, CNTMDOT.)));
SVEPCHGI	Epi/Pandemic Related Change Indicator	Char	NY	CRF Page	Perm	
VISITDY	Planned Study Day of Visit	Num		Protocol	Perm	visitdy = input(visit, ??visitdy.);
TAETORD	Planned Order of Element within Arm	Num		Assigned	Perm	%get_epoch
EPOCH	Epoch	Char	EPOCH	Assigned	Perm	%get_epoch
SVSTDTC	Start Date/Time of Visit	Char	ISO 8601	CRF Page	Exp	
SVENDTC	End Date/Time of Visit	Char	ISO 8601	CRF Page	Exp	
SVSTDY	Study Day of Start of Visit	Num		Derived	Perm	%get_dy
SVENDY	Study Day of End of Visit	Num		Derived	Perm	%get_dy
SVUPDES	Description of Unplanned Visit	Char		CRF Page	Perm	

Table 4. An Example of SV Specification

INTRODUCTION TO *%SV_CODE_GENERATOR* FOR SV DOMAIN AUTOMATION

A deep understanding of SV programming leads us to develop two steps for the automatic SV programming process. The first step focuses on mapping visit-related raw data and calls ***%SDTM_Code_Generator***. The second step focuses on the derivation identified in Steps 1, 2, 3, 5, 6, 7, and 9 of Display 1 and is handled by an additional SAS-based macro named ***%SV_Code_Generator***, which also combines the SAS code from the mapping step and the derivation step to automatically generate SV.sas. This process uses the master-annotation spreadsheet in Table 2 and SV domain specification in Table 4 and follows the 10 sequential steps from Display 1 and Figure 1.

The following sections will explain in detail the automations for Steps 1, 2, 3, and 9 only. Steps 5, 6, and 7 are "standard" programming, which means that they do not change from study to study. Steps 4, 8, and 10 come from the output of ***%SDTM_Code_Generator***. Appendix 3 contains the generated code and corresponding annotations for all the steps. Figure 2 below shows the schematic diagram of how the two macros work together to implement the 10 sequential steps of the SV programming process from Display 1 and Figure 1.

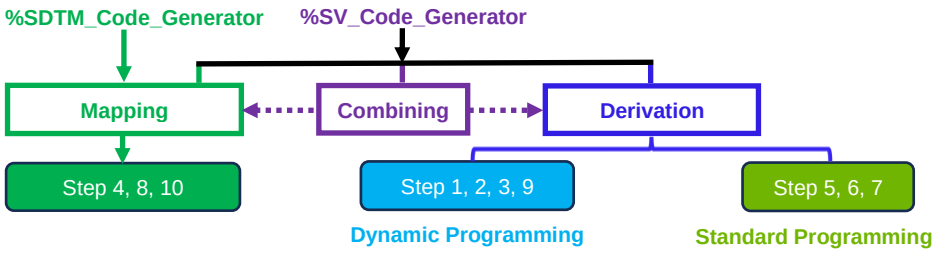


Figure 2. Schematic Diagram of *%SDTM_Code_Generator* and *%SV_Code_Generator*

AUTOMATION STEP 1: IDENTIFY RAW DATASETS WITH BOTH DATES/TIMES AND CLINICAL VISITS

%SV_Code_Generator automatically identifies which raw datasets should be used as the inputs for deriving SVSTDTC and SVENDTC in Step 1.

As introduced in an earlier section ("*Rationale for the Automatic Creation of the SV Domain*"), one of our studies has 75 raw datasets with 143 total date/time variables. Table 5 shows a sample of these raw datasets and their date/time variables. Of note, the last column VARIABLE NAME 2 is derived from SDTM VARIABLE and facilitates the derivation inside the macro.

EDC DATASET NAME	EDC DATASET LABEL	VARIABLE NAME	VARIABLE LABEL	VARIABLE TYPE	SDTM DOMAIN	SDTM VARIABLE	VARIABLE NAME 2
AE	Adverse Events	AEENDAT	End Date	Date	AE	AEENDTC	AEEN
AE	Adverse Events	AEENTIM	End Time	Time	AE	AEENDTC	AEEN
AE	Adverse Events	AESTDAT	Start Date	Date	AE	AESTDTC	AEST
AE	Adverse Events	AESTTIM	Start Time	Time	AE	AESTDTC	AEST
EG	12- Lead ECG - Single Timepoint	EGDAT	Date of the ECG	Date	EG	EGDTC	EG
EG	12- Lead ECG - Single Timepoint	EGTIM	Time of ECG	Time	EG	EGDTC	EG
CLLSLL	Genetic Abnormalities - CLL/SLL	CLLSLLADAT	Assessment Date	Date	FA	FADTC	CLLSLLA
CLLSLL	Genetic Abnormalities - CLL/SLL	CLLSLLCDAT	Collection Date	Date	FA	FADTC	CLLSLLC
CLLSLL	Genetic Abnormalities - CLL/SLL	CLLSLLDAT	Assessment Date	Date	FA	FADTC	CLLSLL
LBBC	Local Lab - B-Cell	LBDAT	Date of Collection	Date	LB	LBDTC	LB
LBCHEM	Local Lab - Chemistry	LBDAT	Date of Collection	Date	LB	LBDTC	LB
LBCOAG	Local Lab - Coagulation	LBDAT	Date of Collection	Date	LB	LBDTC	LB
SV	Subject Visit	VISDAT	Date of Visit	Date	SV	VISDTC in SUPPSV	VIS
VS	Vital Signs	VSDAT	Date of Collection	Date	VS	VSDTC	VS

Table 5. Sample of Raw Datasets With Date/Time Variables From Master-Annotation

From SDTM specifications, ***%SV_Code_Generator*** automatically identifies 15 domains (EG, EX, FA, IE, LB, PC, PE, PR, QS, RS, SS, SV, TR, TU, VS) that contain the variable VISIT.

After restricting the 75 raw datasets to only those who are mapped to the identified 15 SDTM domains above, 66 raw datasets with dates/times and visits (shown in the "select" statement in Display 2) are included in the derivation of SVSTDTC and SVENDTC while 9 raw datasets (shown in Table 6) are excluded as they are mapped to SDTM domains that do not contain VISIT.

EDC DATASET NAME	EDC DATASET LABEL	VARIABLE NAME	VARIABLE LABEL	SDTM DOMAIN	SDTM VARIABLE
AE	Adverse Events	AEENDAT	End Date	AE	AEENDTC
CE	Clinical Events - Signs and Symptoms	CEENDAT	End Date	CE	CEENDTC
CM	Concomitant/Prior Medications	CMENDAT	End Date	CM	CMENDTC
EN	Enrollment	ENRDAT	Enrollment Date	DS	DSSTDTC
EOS	End of Study	EOSDEADT	Death Date	DM	DTHDTC
HOSP	Healthcare Encounters	HOSPENDAT	Utilization End Date	HO	HOENDTC
IC	Informed Consent	ICDAT	Date of Informed Consent	DM	RFICDTC
MH	Medical History	MHENDAT	End Date	MH	MHENDTC
PRM	Public Record Mortality	PRMDAT	Date of subject death per public records	DM	DTHDTC

Table 6. Raw Datasets That Contain Dates But Not Clinical Visits

The following SAS code in Display 4 illustrates how the macro automatically selects raw datasets with both dates/times and clinical visits. It uses both the master-annotation in Table 2 and the SDTM specifications as the inputs. The SAS dataset named as **edc_for_sv** contains the identified raw dataset names and their date/time variable names. The SAS dataset named as **namelist** contains the names of all unique raw datasets with both dates/times and clinical visits as well as other raw datasets mapped to the SV domain (e.g., NV: "Next Visit"). ***%SV_Code_Generator*** then does simple data manipulations to **namelist** and generates the SAS dataset for the PROC DATASETS step shown in Display 2.

```
*** get the EDC dataset names with date variables from master-annotation;
proc sort data=other.master out=master_dtc;by sdtm_domain variable_name;
     where strip(variable_type) in ('Date','Time') or strip(sdtm_domain)='SV';
run;
*** from SDTM specs to get SDTM domains with VISIT;
proc sort data=sdtmspec.all_vars out=spec_wvisit(rename=(domain=sdtm_domain)) nodupkeys;
     by domain;
     where strip(variable)='VISIT';
run;
*** only select edc datasets mapped into SDTM domains with 'VISIT';
data edc_for_sv;
     merge master_dtc(in=a) spec_wvisit(in=b keep=sdtm_domain);
     by sdtm_domain;
     if a and b;
run;
```

Display 4. SAS Code Identifying Raw Datasets With Both Dates/Times and Clinical Visits

AUTOMATION STEP 2: SEPARATE THEM INTO TWO DATA BLOCKS: ONE WITH DATES ONLY, ANOTHER WITH BOTH DATES AND TIMES

%SV_Code_Generator automatically separates the raw datasets with dates/times and visits into two data blocks: one containing only date variables and another containing both date and time variables.

The following SAS code in Display 5 illustrates this separation utilizing *VARIABLE_TYPE* from the master-annotation spreadsheet in Table 2.

```
*** get the edc names with variable type='Date';
proc sort data=edc_for_sv out=edcnm_date nodupkeys;by edc_dataset_name var_nm2 variable_name;
      where variable_type='Date';
run;
*** get the edc names with variable type='Time';
proc sort data=edc_for_sv out=edcnm_time nodupkeys;by edc_dataset_name var_nm2 variable_name;
     where variable_type in ('Time');
run;
*** separate them into two datasets: one with date only, another with both date and time;
data edcnm_wotm
     edcnm_wtm;
     merge edcnm_date(in=a) edcnm_time(in=b keep=edc_dataset_name var_nm2 variable_name
                                       rename=(variable_name=variable_name_tm));
     by edc_dataset_name var_nm2;
     if a and not b then output edcnm_wotm;
     else if a and b then output edcnm_wtm;
run;
```

Display 5. SAS Code Identifying Raw Datasets With Only Date Variables and Raw Datasets With Both Date and Time Variables

Table 7 below shows the 9 raw datasets with both date and time variables mapped into SDTM domains that contain VISIT as identified by the macro.

EDC DATASET NAME	EDC DATASET LABEL	VARIABLE NAME (DATE)	VARIABLE NAME (TIME)	VARIABLE NAME 2	SDTM VARIABLE	SDTM DOMAIN
BIO	Exploratory Biomarkers - Infusion Day	BIOPOSTDAT	BIOPOSTTIM	BIOPOST	LBDTC	LB
BIO	Exploratory Biomarkers - Infusion Day	BIOPREDAT	BIOPRETIM	BIOPRE	LBDTC	LB
BIONON	Exploratory Biomarkers	NONBIODAT	NONBIOTIM	NONBIO	LBDTC	LB
CY	Cytokines	CYKDAT	CYKTIM	CYK	LBDTC	LB
EG	12- Lead ECG - Single Timepoint	EGDAT	EGTIM	EG	EGDTC	EG
EX1	Lymphodepleting Chemotherapy: Fludarabine	EX1ENDAT	EX1ENTIM	EX1EN	EXENDTC	EX
EX1	Lymphodepleting Chemotherapy: Fludarabine	EX1STDAT	EX1STTIM	EX1ST	EXSTDTC	EX
EX2	Lymphodepleting Chemotherapy: Cyclophosphamide	EX2ENDAT	EX2ENTIM	EX2EN	EXENDTC	EX
EX2	Lymphodepleting Chemotherapy: Cyclophosphamide	EX2STDAT	EX2STTIM	EX2ST	EXSTDTC	EX
EX3	Treatment Dosing	EX3ENDAT	EX3ENTIM	EX3EN	EXENDTC	EX
EX3	Treatment Dosing	EX3STDAT	EX3STTIM	EX3ST	EXSTDTC	EX
PK	Treatment PK	PKDAT	PKTIM	PK	PCDTC	PC
PKINF	Treatment PK-Infusion Day	PKPREDAT	PKPRETIM	PKPRE	PCDTC	PC

Table 7. An Example of 9 Raw Datasets With Both Date and Time Variables Mapped into SDTM Domains That Contain VISIT

AUTOMATION STEP 3: FOR EACH DATA BLOCK, STACK EACH DATASET AND STANDARDIZE ITS DATE VARIABLE NAME INTO A COMMON NAME: SVDTC

Following Step 2, ***%SV_Code_Generator*** automatically generates SAS code for the first part of two data steps ("data try1" and "data try2" blocks in Appendix 3), which stack all raw datasets with date variables only and all raw datasets with both date and time variables, respectively, standardize their variable names into common names: SVDT and SVTM, and convert them to SVDTC.

Display 6 shows the SAS code for using the dataset EDCNM_WOTM from Step 2 to generate macro variables for the raw dataset names with dates only. Macro variables are created for each raw dataset name, variable name, and the total number of raw datasets. These macro variables are then used inside a macro %doit1() shown in Display 7.

```
** get each EDC dataset name and its variable name for date;
proc sort data=edcnm_wotm;by edc_dataset_name variable_name;run;
%let noutsv1=0;
data _null_;
     set edcnm_wotm end=eof;
     by edc_dataset_name variable_name;
     call symput('edcname'||strip(put(_n_,best.)),strip(edc_dataset_name));** each EDC dataset name;
     ** each variable name of each EDC dataset name for 'date';
     call symput('varname'||strip(put(_n_,best.)),strip(variable_name));
     if eof then call symput('noutsv1',strip(put(_n_,best.)));** number of EDC datasets;
run;
```

Display 6. SAS Code Creating Macro Variables for Raw Dataset Names, Variable Names, and the Total Number of Raw Datasets From EDCNM_WOTM in Step 2

Display 7 below shows a macro generating a SAS dataset named as *_try1head*, which stacks all raw datasets with dates only and renames their date variable names to SVDT. It uses a macro %DO loop to stack the datasets and Lines 178 and 182 standardize these date variable names to a common name: SVDT.

The data step "**data try1;**" in Appendix 3 shows the SAS code generated from the macro call of %doit1() to stack 57 raw datasets and rename their date variable names to SVDT. SVDT has a numeric date format and is then converted to SVDTC, which has a character ISO 8601 format.

```
166 %macro doit1();
167 data _try1head;
168      length lines $200.;
169      subseq=0.01;lines='*** Below is the SAS codes for the EDC datasets only with dates;';output;
170      subseq=0.1;lines='data try1;';output;
171      subseq=0.2;lines="     attrib "||"&"||"attrib.;";output;
172      subseq=0.25;lines="     length UNSAPERF $200. SVDTC $20.;";output;
173      subseq=0.3;lines='     set ';output;
174      %do i=1 %to &noutsv1;
175          subseq=%sysevalf(0.3+0.001*&i);
176          %if &i<&noutsv1 %then %do;
177              %if &&edcname&i=UV or &&edcname&i=SV %then %do;
178                 lines='          '||"&&edcname&i.(drop=studyid siteid rename=(&&varname&i.=SVDT) in=_&&edcname&i.)";%end;
179              %else %do;lines='          '||"&&edcname&i.(drop=studyid siteid rename=(&&varname&i.=SVDT))";%end;
180              output;
181          %end;
182          %else %do;lines='          '||"&&edcname&i.(drop=studyid siteid rename=(&&varname&i.=SVDT));";output;%end;
183      %end;
184 run;
185 %mend;
```

Display 7. A SAS Macro Generating a SAS Dataset (_try1head) to Stack Raw Datasets and Rename Their Date Variable Names to SVDT

Similarly, Displays 8 and 9 show the same programming logic for using dataset EDCNM_WTM from Step 2 to stack all raw datasets with both date and time variables and rename their date variable names to SVDT and time variable names to SVTM. The SAS dataset is named as *_try2head*.

```
%let noutsv2=0;
data _null_;
     set edcnm_wtm end=eof;
     by edc_dataset_name variable_name;
     call symput('_edcname'||strip(put(_n_,best.)),strip(edc_dataset_name));
     call symput('_varname'||strip(put(_n_,best.)),strip(variable_name));
     call symput('_varname2'||strip(put(_n_,best.)),strip(variable_name_tm));
     if eof then call symput('noutsv2',strip(put(_n_,best.)));
run;
```

Display 8. SAS Code Creating Macro Variables for Raw Dataset Names, Variable Names, and the Total Number of Raw Datasets From EDCNM_WTM in Step 2

```
329 %macro doit2();
330 data _try2head;
331      length lines $200.;
332      section=2;
333      subseq=0.01;lines='*** Below is the SAS codes for the EDC datasets with date and time;';output;
334      subseq=0.1;lines='data try2;';output;
335      subseq=0.2;lines="     attrib "||"&"||"attrib.;";output;
336      subseq=0.25;lines="     length SVDTC $20.;";output;
337      subseq=0.3;lines='     set  ';output;
338      %do i=1 %to &noutsv2;
339          subseq=%sysevalf(0.3+0.001*&i);
340          %if &i<&noutsv2 %then %do;
341              lines='          '||"&&_edcname&i(drop=studyid siteid rename=(&&_varname&i=SVDT &&_varname2&i.=SVTM))";
342               output;
343          %end;
344          %else %do;
345               lines='          '||"&&_edcname&i(drop=studyid siteid rename=(&&_varname&i=SVDT &&_varname2&i.=SVTM));";
346               output;
347          %end;
348       %end;
349 run;
350 %mend;
```

Display 9. A SAS Macro Generating a SAS Dataset (_try2head) to Stack Raw Datasets and Rename Their Date Variable Names to SVDT and Time Variable Names to SVTM

The data step "**data try2;**" in Appendix 3 shows SAS code generated from the macro call of %doit2() to stack 9 raw datasets with both date and time variables and rename their date variable names to SVDT and time variable names to SVTM. SVDT and SVTM are then combined into a single SVDTC, which has a character ISO 8601 format.

From the example above in Appendix 3, the manual development of SAS codes to stack all raw datasets and standardize their date/time variable names to SVDTC is time-consuming and error prone. The raw data with dates/times and clinical visits collected in a study change from study to study, which has a negative impact on developing SAS codes for SV domain. The changes of date variable and/or time variable names of these raw datasets from the EDC database build make it even worse. Steps 1-3 automate the programming to achieve huge time saving and efficiency along with the guarantee of high quality. It is the main contribution to SV domain automation!

AUTOMATION STEP 9: MAP OTHER RAW DATASET VARIABLES INTO SUPPSV

SUPPSV is automatically generated by two steps. The first step involves calling ***%SDTM_Code_Generator(domain_=SV)***, which generates SAS code that contains the data step for SUPPSV. Please refer to Block 4 in Appendix 2 as an example. Among these 7 blocks of SAS codes for each QNAM, one block (QNAM = "UNSAPERF") requires preprocessing before it can be included in SUPPSV. The Unscheduled Subject Visit form (Appendix 1) has separate fields to indicate whether a specific assessment was performed at an unscheduled visit. In the preprocessing, these separate raw dataset variables (annotated in blue in Appendix 1) are concatenated into a single variable with a comma as the delimiter and then mapped to QVAL with QNAM = "UNSAPERF" (annotated in red in Appendix 1). Table 8 shows an example of what the final output looks like.

STUDYID	RDOMAIN	USUBJID	IDVAR	IDVARVAL	QNAM	QLABEL	QVAL	QORIG
Project-Study-101	SV	Project-Study-101-101-002	VISITNUM	1.01	UNSAPERF	Unscheduled Assessments Performed	EG, VS	CRF
Project-Study-101	SV	Project-Study-101-101-002	VISITNUM	1.02	UNSAPERF	Unscheduled Assessments Performed	CHEM, HEM, VS	CRF
Project-Study-101	SV	Project-Study-101-101-002	VISITNUM	9.01	UNSAPERF	Unscheduled Assessments Performed	VS	CRF

Table 8. An Example of SUPPSV.QVAL where QNAM = "UNSAPERF"

"_BLOCK A" in Appendix 3 shows the SAS code for concatenating 47 raw dataset variables into a single variable UNSAPERF. The ***%SV_Code_Generator*** macro automatically retrieves these variable names from the **master-annotation** (Table 2). Display 10 below shows the SAS code used to retrieve these variable names. Lines 3 and 4 show that the master-annotation is the unique source of the programming and TRT ASSIGN = "COMBINE" flags the variables for preprocessing at the start of the automation of the SV domain.

```
1 data supp_
2      unspr;
3      set other.master(where=(strip(suppdomain)^='' and sdtm_domain='SV'));
4      if trt_assign='COMBINE' and qnam='UNSAPERF' then output unspr;
5      else output supp_;
6 run;
```

Display 10. SAS Code to Retrieve the Variable Names for the Concatenation Specified in the Master-Annotation for the SV Domain

Once the variable names are retrieved, the macro concatenates them with a comma as the delimiter. Display 11 contains the SAS code that automates this step. The generated code is then output into SV.sas and can be seen in "_BLOCK A" in Appendix 3. Afterwards, the macro retrieves the SAS code for mapping the concatenated variable UNSAPERF to QVAL with QNAM = "UNSAPERF" in the data step for SUPPSV ("_BLOCK C" in Appendix 3).

```
data unspr2;
     length lines $200.;
     set unspr end=eof;
     section=1;
     subseq=20+_n_/10;
     lines='          '||"if _UV and " || strip(variable_name) || "=1 then " || strip(qnam)
          || " = catx(', '," || strip(qnam) || ", '" || strip(variable_name) || "');";
     output;
     if eof then do;
          subseq=30;lines='   ';output;
          subseq=31;lines='          '||'in_UV = _UV;';output;
          subseq=32;lines='          '||'in_SV = _SV;';output;
          subseq=33;lines='   ';output;
          subseq=34;
          lines='          '||'if _SV or not missing(SVDTC) then output;/** with missing date from scheduled visits **/';
          output;
          subseq=35;lines='run;';output;
          subseq=36;lines='   ';output;
          subseq=20;lines='          '||'call missing(UNSAPERF);';output;
     end;
run;
```

Display 11. SAS Code to Concatenate Variable Names Specified in the Master-Annotation for the SV Domain

GENERATING SV.SAS

The last four sections above introduced how ***%SV_Code_Generator*** automatically generates the SAS code for the derivations from Steps 1, 2, 3, and 9 of Display 1. For the derivations from Steps 5, 6, and 7, the macro simply outputs each line of SAS code shown in _BLOCK B of Appendix 3. The remaining Steps 4, 8, and 10 are from the output of ***%SDTM_Code_Generator***. ***%SV_Code_Generator*** follows the general SV programming logic flow to assemble these derivations and mappings into the final SAS code for the SV domain, which is depicted by Figure 2. This generated code is then saved as a SAS dataset prior to being output as SV.sas using a simple SAS data step as shown below in Display 12.

```
825 data _null_;
826      set mainf2;
827      file "&outdir./SV.sas";
828      put @1 lines $255.;
829 run;
```

Display 12. SAS Data _NULL_ Step to Output a SAS Program: SV.sas

HOW TO GUARANTEE ALL RAW DATASET VARIABLES ARE MAPPED INTO THE SV DOMAIN

[1] demonstrates how ***%SDTM_Code_Generator*** automatically detects any raw dataset variables unmapped in SDTM. Once the errors from omissions of raw dataset variables for SDTM are detected, they can be easily fixed. Hence, this step guarantees all raw dataset variables are accounted for in SDTM programming.

The first six columns of the master-annotation in Table 2 are automatically created from the CRF specification. For SV, these columns contain all raw dataset variables collected for both scheduled visits and unscheduled visits. As these come directly from the CRF specification, there is no human error. The first part of Table 2, where TRT ASSIGN = Blank, is covered by ***%SDTM_Code_Generator*** to guarantee that these raw dataset variables are accounted for SV domain. The second part, where TRT ASSIGN = “COMBINE”, contains the variables which require concatenation prior to mapping into QVAL. The SAS code for processing and mapping these raw dataset variables are automatically generated by ***%SDTM_Code_Generator*** and ***%SV_Code_Generator***. Furthermore, as discussed in the section titled “*Automation Step 1: Identify Raw Datasets With Both Dates/Times and Clinical Visits*”, ***%SV_Code_Generator*** uses the master-annotation and SV domain specification to automatically identify raw datasets and variables that should be used in the derivation of SVSTDTC and SVENDTC.

Hence, the sequential programming automations (e.g., automatic generation of the first six columns of the master-annotation, automatic detection of raw dataset variable omissions for SDTM by ***%SDTM_Code_Generator***, and automatic preparations of Steps 1 and 9 by ***%SV_Code_Generator***) guarantees all raw dataset variables are accounted for in SV domain programming.

HOW TO HANDLE EXTERNAL DATASETS

The external datasets (such as central safety labs, biomarkers, etc.) are specified by data transfer specifications from different vendors. Because they often contain visit-dependent data, they should also be included in the derivation of the SV dataset.

Both ***%SDTM_Code_Generator*** and ***%SV_Code_Generator*** are developed to map the raw data collected per CRFs to SDTM domains. The latter follows the same logic as the former in terms of handling external datasets. The rationale is fully explained in [1]. The flexibility is built into the former, and it is quoted as follows:

*"When the external datasets are ready for inclusion, the team can decide if the new programming should be added to either **%SDTM_Code_Generator** or the related individual SDTM SAS program. The decision requires balancing the generalization of the macro for future use with the spending of more time/resources in updating the macro and its potential impact of timelines"* [1].

%SV_Code_Generator has been built to generate a SAS program comment to suggest adding extra lines of SAS codes to handle the external data once they are ready to be included at a later time in the programming development cycle (e.g., before FDA submission). Display 13 shows the message along with the annotation for Step 4.5.

```
/*read in external data, and combine with try3, eg., central tumor data, etc.*/
```

Step 4.5: Bring External Datasets and Combine Them with These Two Data Blocks Above if There Exists Any

Display 13. A SAS Program Comment for Handling External Data in Step 4.5 of Appendix 3.

The team can make the decision on whether the macro should be updated or whether SV.sas should be updated to account for the new external dataset(s). Due to the dissimilarity of external dataset(s) and their metadata and the unpredictable timing of data availability, the later manual updating of SV.sas is probably the better solution to meet timelines and achieve efficiency.

INTRODUCTION TO OUR VALIDATION PROCESS FOR SV PROGRAMMING

For SV domain validation, we follow the same process as the other SDTM domains [1], which is quoted below along with Figure 3 for ease of access:

"Our SDTM programming validation consists of the following three steps: code reviewing, real data testing, and developing independent mapping SAS programs to validate relatively complicated SDTM datasets as needed per the team's decision. This validation process validates both the macro and each SDTM mapping SAS program. [Figure 3] below depicts the new validation process" [1].

The first two steps, code reviewing and real data testing, are strictly followed until the user ensures that the SAS program is thoroughly tested and meets the requirements for the SV domain when no external data are available. Once the external data are included, the team decides if code reviewing and real data testing are sufficient and acceptable for fully validating the new standalone programming section added for external dataset(s) inside SV.sas. If not, the validator should independently develop a SAS program to fully validate the external data programming for SV.

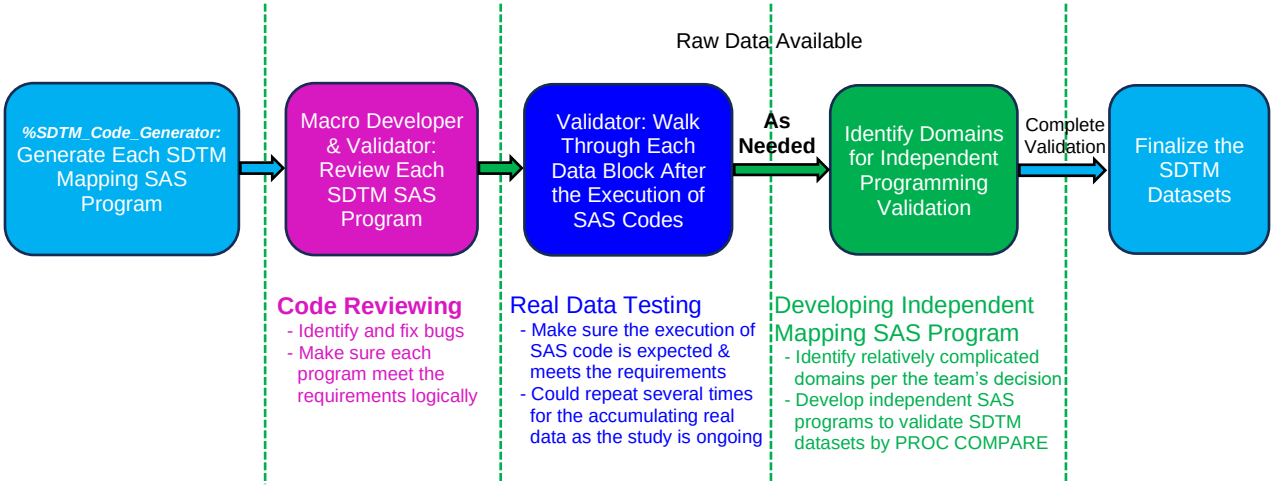


Figure 3. The Logic Flow of Our SDTM Programming Validation Process [1]

CONCLUSION

This paper introduced a new approach to automatic SAS code generation for the Subject Visits (SV) SDTM domain. Automation is achieved by using a metadata-driven method that leverages CRF specifications and SDTM standards through our two macros: ***%SDTM_Code_Generator*** and ***%SV_Code_Generator***. Our SV programming process is an expansion of the SDTM automation we developed with ***%SDTM_Code_Generator*** [1], and it inherits the same practicality, efficiency, flexibility, transparency, and scalability. We also demonstrated how to process the "dynamic" parts of SV programming, how to account for all relevant variables, how to deal with complex derivations, how to handle the external datasets, and how to validate the SV dataset generated by the new automation process.

The sharing of hands-on experience in this paper is to assist readers with applying this methodology to generate the SV domain in the early stages of clinical reporting with the guarantee of technical accuracy, in addition to cost-effectiveness and efficiency. These automations can also be used to expedite and assist with clinical data review and data cleaning efforts efficiently and economically.

REFERENCES

[1] Xiangchen (Bob) Cui; Min Chen; Jessie Wang. A Practical Approach to Automating SDTM Using a Metadata-Driven Method That Leverages CRF Specifications and SDTM Standards. PHUSE US Connect 2024 and PharmaSUG 2024

[2] Cleopatra DeLeon & Laura Bellamy. Dynamically harvesting study dates to construct & QC the SV SDTM domain. PHUSE US Connect 2019

[3] CDISC Study Data Tabulation Model and SDTMIG v3.4 at http://www.cdisc.org/sdtm

[4] FDA Validator Rules. December 2022. Available at https://www.fda.gov/industry/fda-data-standards-advisory-board/study-data-standards-resources

[5] [U.S. Department of Health and Human Services, Food and Drug Administration, Study Data Technical Conformance Guide: Technical Specifications Document. June 2023. Available at https://www.fda.gov/media/153632/download

[6] Henry B. Winsor, and Mario Widel, "CREATING SV AND SE FIRST", PharmaSUG 2010

[7] Xiangchen (Bob) Cui; Scott Moseley; Min Chen. A Cost-Effective SDTM Conversion for NDA Electronic Submission. Proceedings of the Pharmaceutical SAS® Users Group Conference, PharmaSUG 2011

ACKNOWLEDGMENTS

Appreciation goes to PK Morrow for her invaluable review and comments.

CONTACT INFORMATION

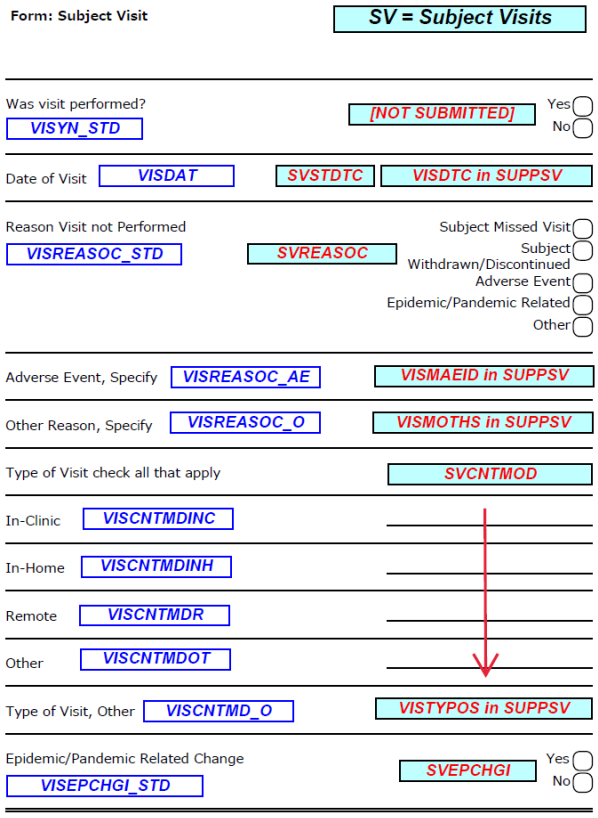
Your comments and questions are valued and encouraged. Contact the author at:

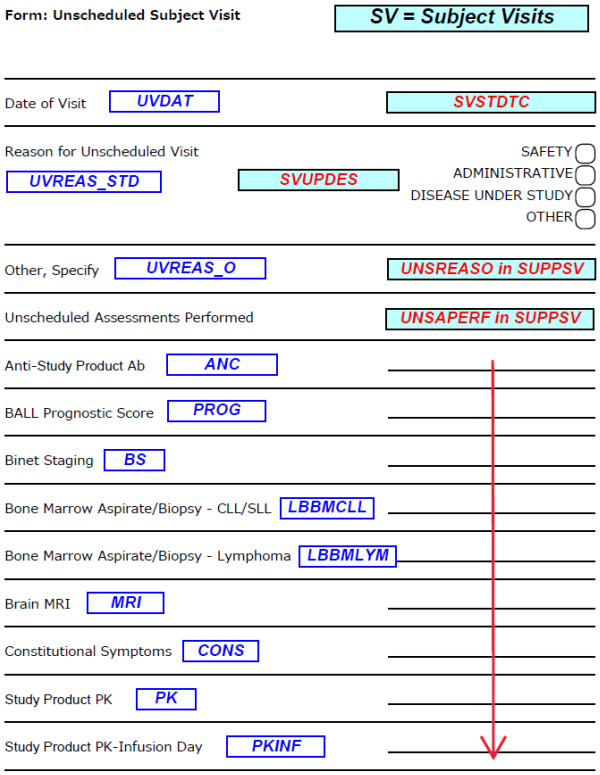
Name: Xiangchen (Bob) Cui, Ph.D.
Enterprise: CRISPR Therapeutics AG
Address: 105 West 1st Street
City, State ZIP: Boston, MA 02127
Work Phone: 908-240-4086
E-mail: xiangchen.cui@crisprtx.com

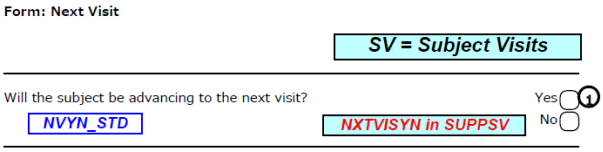
Name: Jessie Wang
Enterprise: CRISPR Therapeutics AG
Address: 105 West 1st Street
City, State ZIP: Boston, MA 02127
Work Phone: 214-668-2107
E-mail: jessie.wang@crisprtx.com

Name: Min Chen
Enterprise: CRISPR Therapeutics AG
Address: 105 West 1st Street
City, State ZIP: Boston, MA 02127
Work Phone: 857-928-4347
E-mail: min.chen@crisprtx.com

APPENDIX 1. A SAMPLE OF THE ANNOTATED CRFS CONTAINING BOTH RAW DATASET VARIABLE NAMES AND THEIR MAPPED SDTM VARIABLES AND SUPPLEMENTAL QUALIFIERS FOR THE SUBJECT VISIT (SV), UNSCHEDULED SUBJECT VISIT (UV), AND NEXT VISIT (NV) FORMS







APPENDIX 2. AN EXAMPLE OF _SV.SAS FROM CALLING *%SDTM_CODE_GENERATOR*

```
data try;
     attrib &attrib.;
     set
         NV(drop=studyid siteid in=_NV)
         SV(drop=studyid siteid in=_SV)
         UV(drop=studyid siteid in=_UV);

         STUDYID = 'Project-Study-101';
         DOMAIN = 'SV';
         USUBJID = strip(STUDYID)||strip(substr(SUBJECT,4));
         visit = strip(put(folder, $visit.));
         visitnum = input(visit, ??visitnum.);
         if VISIT ^= 'Unscheduled' then SVPRESP = 'Y';
         visitdy = input(visit, ??visitdy.);
         if not missing(VISREASOC_STD) then SVREASOC=strip(VISREASOC_STD);
         SVCNTMOD = strip(coalescec(put(VISCNTMDINC, CNTMDINC.), put(VISCNTMDINH, CNTMDINH.),
                                    put(VISCNTMDR, CNTMDR.), put(VISCNTMDOT, CNTMDOT.)));
         if not missing(VISEPCHGI_STD) then SVEPCHGI=strip(VISEPCHGI_STD);
         if not missing(UVREAS_STD) then SVUPDES=strip(UVREAS_STD);
         if not missing(UVDAT) then SVSTDTC=strip(put(UVDAT/24/3600,yymmdd10.));
run;
*********  Programming Note: SDTM Variable: SVSTDY Needs the Derivation by the Macro Call: %get_dy;
%get_dy(_DATEVAR=SVSTDTC,_DAYVAR=SVSTDY);
*********  Programming Note: SDTM Variable: SVENDY Needs the Derivation by the Macro Call: %get_dy;
%get_dy(_DATEVAR=SVENDTC,_DAYVAR=SVENDY);
proc sort data=try;by studyid usubjid VISITNUM;run;
*** Output permanent datasets to: wsdtm ***;
data wsdtm.&domain.(keep=&keep label=&label);
       attrib &attrib.;
       set try;
       format _all_;
       informat _all_;
run;
data supp&domain.;
     attrib &attrib_supp.;
     set try;
     rdomain = "&domain.";
     idvar = 'VISITNUM';
     idvarval = strip(put(VISITNUM, best.));
     if not missing(NVYN_STD) then do;
          qnam='NXTVISYN';
          qlabel='Subject Advancing to Next Visit?';
          qval=strip(NVYN_STD);
          qorig='CRF';
          qeval='';
          output;
     end;
     if not missing(VISDAT) then do;
          qnam='VISDTC';
          qlabel='Visit Date';
          qval=strip(put(VISDAT/24/3600,yymmdd10.));
          qorig='CRF';
          qeval='';
          output;
     end;
     if not missing(UVREAS_O) then do;
          qnam='UNSREASO';
          qlabel='Unscheduled Visit: Other Reason, Specify';
          qval=strip(UVREAS_O);
          qorig='CRF';
          qeval='';
          output;
     end;
     if not missing(VISREASOC_AE) then do;
          qnam='VISMAEID';
          qlabel='Missed Visit: Adverse Event, Specify';
          qval=strip(VISREASOC_AE);
          qorig='CRF';
          qeval='';
          output;
     end;
     if not missing(UNSAPERF) then do;
          qnam='UNSAPERF';
          qlabel='Unscheduled Assessments Performed';
          qval=strip(UNSAPERF);
          qorig='CRF';
          qeval='';
```

Block 1

Block 2

Block 3

Block 4

Variable **UNSAPERF** is derived from extra programming to concatenate the values of multiple raw dataset variables by ",". This concatenation is done by ***%SV_Code_Generator***, and the output code is in _BLOCK A of Appendix 3.

```
          output;
     end;
     if not missing(VISREASOC_O) then do;
          qnam='VISMOTHS';
          qlabel='Missed Visit: Other Reason, Specify';
          qval=strip(VISREASOC_O);
          qorig='CRF';
          qeval='';
          output;
     end;
     if not missing(VISCNTMD_O) then do;
          qnam='VISTYPOS';
          qlabel='Type of Visit: Other, Specify';
          qval=strip(VISCNTMD_O);
          qorig='CRF';
          qeval='';
          output;
     end;
run;
```

Block 4

```
proc sort data=&suppdomain.;by usubjid idvarval qnam;run;
data wsdtm.&suppdomain.(keep=&keep_supp label=&label_supp);
       attrib &attrib_supp.;
       set &suppdomain.;
       format _all_;
       informat _all_;
run;
```

Block 5

APPENDIX 3. AN EXAMPLE OF SV.SAS FROM CALLING *%SV_CODE_GENERATOR*

Step 1: Identify Raw Datasets with Both Dates/Times and Clinical Visits

```
proc datasets lib=work kill nolist nowarn;
     copy in=raw out=work;
          select ANC BIO BIONON BS CLLSLL CONS CP CY DIA ECHO ECOG EG EQ5D5L EX1 EX2 EX3
                 FACT FACTLEU ICE INL INTL1 INTL2 ITL1 ITL2 IWCLL LBBC LBBMCLL LBBMLYM LBCHEM LBCOAG LBCRS LBHM
                 LBIMMU LBLYM LBPREG LBSER LP LS LUGANO MR MRD MRI NL NTL1 NTL2 NV ORG PE PET1 PET2
                 PK PKINF PROG RADPOST RADPRE RETX SCTPOST SCTPRE SS SV TB THERPOST THERPRE TL1 TL2 UV VS;
     copy in=sdtm out=work;
          select dm;
quit;

*** Below is the SAS codes for the EDC datasets only with dates;
data try1;
     attrib &attrib.;
     length UNSAPERF $200. SVDTC $20.;
     set
          ANC(drop=studyid siteid rename=(ANCDAT=SVDT))
          BS(drop=studyid siteid rename=(BSDAT=SVDT))
          CLLSLL(drop=studyid siteid rename=(CLLSLLADAT=SVDT))
          CLLSLL(drop=studyid siteid rename=(CLLSLLCDAT=SVDT))
          CLLSLL(drop=studyid siteid rename=(CLLSLLDAT=SVDT))
          CONS(drop=studyid siteid rename=(CONSYMDAT=SVDT))
          CP(drop=studyid siteid rename=(CPENDAT=SVDT))
          CP(drop=studyid siteid rename=(CPSTDAT=SVDT))
          DIA(drop=studyid siteid rename=(DIADAT=SVDT))
          ECHO(drop=studyid siteid rename=(ECHODAT=SVDT))
          ECOG(drop=studyid siteid rename=(ECOGDAT=SVDT))
          EQ5D5L(drop=studyid siteid rename=(EQ5D5LDAT=SVDT))
          FACT(drop=studyid siteid rename=(FACTDAT=SVDT))
          FACTLEU(drop=studyid siteid rename=(FACTLDAT=SVDT))
          ICE(drop=studyid siteid rename=(IADAT=SVDT))
          INL(drop=studyid siteid rename=(INLMDAT=SVDT))
          INTL1(drop=studyid siteid rename=(INTL1DAT=SVDT))
          INTL2(drop=studyid siteid rename=(INTL2DAT=SVDT))
          ITL1(drop=studyid siteid rename=(ITL1DAT=SVDT))
          ITL2(drop=studyid siteid rename=(ITL2DAT=SVDT))
          IWCLL(drop=studyid siteid rename=(IWCLLDAT=SVDT))
          LBBC(drop=studyid siteid rename=(LBDAT=SVDT))
          LBBMCLL(drop=studyid siteid rename=(BMADAT_CLL=SVDT))
          LBBMCLL(drop=studyid siteid rename=(BMBDAT=SVDT))
          LBBMLYM(drop=studyid siteid rename=(BMADAT_LYM=SVDT))
          LBBMLYM(drop=studyid siteid rename=(BMBDAT=SVDT))
          LBCHEM(drop=studyid siteid rename=(LBDAT=SVDT))
          LBCOAG(drop=studyid siteid rename=(LBDAT=SVDT))
          LBCRS(drop=studyid siteid rename=(LBDAT=SVDT))
          LBHM(drop=studyid siteid rename=(LBDAT=SVDT))
          LBIMMU(drop=studyid siteid rename=(LBDAT=SVDT))
          LBLYM(drop=studyid siteid rename=(LBDAT=SVDT))
          LBPREG(drop=studyid siteid rename=(PGDAT=SVDT))
          LBSER(drop=studyid siteid rename=(LBDAT=SVDT))
```

Step 2: Identify Raw Datasets With Dates Only Along With Clinical Visits

Step 3.1: Stack Raw Datasets and Standardize Their Date Variable Names Into a Common Name (SVDTC)

```
LP(drop=studyid siteid rename=(LPDAT=SVDT))
LS(drop=studyid siteid rename=(LSDAT=SVDT))
LUGANO(drop=studyid siteid rename=(LUGDAT=SVDT))
MR(drop=studyid siteid rename=(MRDAT=SVDT))
MRD(drop=studyid siteid rename=(MRDDAT=SVDT))
MRI(drop=studyid siteid rename=(MRIDAT=SVDT))
NL(drop=studyid siteid rename=(NLMDAT=SVDT))
NL(drop=studyid siteid rename=(NLUDAT=SVDT))
NTL1(drop=studyid siteid rename=(NTL1MDAT=SVDT))
NTL1(drop=studyid siteid rename=(NTL1UDAT=SVDT))
NTL2(drop=studyid siteid rename=(NTL2DAT=SVDT))
NTL2(drop=studyid siteid rename=(NTL2UDAT=SVDT))
ORG(drop=studyid siteid rename=(OGLVDAT=SVDT))
ORG(drop=studyid siteid rename=(OGSPDAT=SVDT))
PE(drop=studyid siteid rename=(PEDAT=SVDT))
PET1(drop=studyid siteid rename=(PET1DAT=SVDT))
PET2(drop=studyid siteid rename=(PET2DAT=SVDT))
PROG(drop=studyid siteid rename=(PROGDAT=SVDT))
RADPOST(drop=studyid siteid rename=(RADPOSTENDAT=SVDT))
RADPOST(drop=studyid siteid rename=(RADPOSTSTDAT=SVDT))
RADPRE(drop=studyid siteid rename=(RADPREENDAT=SVDT))
RADPRE(drop=studyid siteid rename=(RADPRESTDAT=SVDT))
RETX(drop=studyid siteid rename=(RETXDAT=SVDT))
SCTPOST(drop=studyid siteid rename=(SCTALLODAT=SVDT))
SCTPOST(drop=studyid siteid rename=(SCTAUTODAT=SVDT))
SCTPRE(drop=studyid siteid rename=(SCTAUTODAT=SVDT))
SS(drop=studyid siteid rename=(SSDAT=SVDT))
SV(drop=studyid siteid rename=(VISDAT=SVDT) in=_SV)
TB(drop=studyid siteid rename=(TBDAT=SVDT))
THERPOST(drop=studyid siteid rename=(THERPOSTENDDAT=SVDT))
THERPOST(drop=studyid siteid rename=(THERPOSTSTDAT=SVDT))
THERPRE(drop=studyid siteid rename=(THERENDDAT=SVDT))
THERPRE(drop=studyid siteid rename=(THERSTDAT=SVDT))
TL1(drop=studyid siteid rename=(TL1MDAT=SVDT))
TL1(drop=studyid siteid rename=(TL1UDAT=SVDT))
TL2(drop=studyid siteid rename=(TL2MDAT=SVDT))
TL2(drop=studyid siteid rename=(TL2UDAT=SVDT))
UV(drop=studyid siteid rename=(UVDAT=SVDT) in=_UV)
VS(drop=studyid siteid rename=(VSDAT=SVDT));

if folder in ('VI', 'DA') then delete;/* VI=Visit Independent, DA=Disease Assessment */

SVCNTMOD = strip(coalescec(put(VISCNTMDINC, CNTMDINC.), put(VISCNTMDINH, CNTMDINH.),
           put(VISCNTMDR, CNTMDR.),put(VISCNTMDOT, CNTMDOT.)));
if not missing(VISEPCHGI_STD) then SVEPCHGI=strip(VISEPCHGI_STD);
if not missing(UVREAS_STD) then SVUPDES=strip(UVREAS_STD);
if not missing(SVDT) then SVDTC=strip(put(SVDT/24/3600,yymmdd10.));

if _SV then VISDAT = SVDT;
```

From Block 2 of _SV.SAS

```
call missing(UNSAPERF);
if _UV and ANC=1 then UNSAPERF = catx(', ',UNSAPERF, 'ANC');
if _UV and PROG=1 then UNSAPERF = catx(', ',UNSAPERF, 'PROG');
if _UV and BS=1 then UNSAPERF = catx(', ',UNSAPERF, 'BS');
if _UV and LBBMCLL=1 then UNSAPERF = catx(', ',UNSAPERF, 'LBBMCLL');
if _UV and LBBMLYM=1 then UNSAPERF = catx(', ',UNSAPERF, 'LBBMLYM');
if _UV and MRI=1 then UNSAPERF = catx(', ',UNSAPERF, 'MRI');
if _UV and CONS=1 then UNSAPERF = catx(', ',UNSAPERF, 'CONS');
if _UV and PK=1 then UNSAPERF = catx(', ',UNSAPERF, 'PK');
if _UV and PKINF=1 then UNSAPERF = catx(', ',UNSAPERF, 'PKINF');
if _UV and CY=1 then UNSAPERF = catx(', ',UNSAPERF, 'CY');
if _UV and EG=1 then UNSAPERF = catx(', ',UNSAPERF, 'EG');
if _UV and ECOG=1 then UNSAPERF = catx(', ',UNSAPERF, 'ECOG');
if _UV and EQ5D5L=1 then UNSAPERF = catx(', ',UNSAPERF, 'EQ5D5L');
if _UV and BIO=1 then UNSAPERF = catx(', ',UNSAPERF, 'BIO');
if _UV and BIONON=1 then UNSAPERF = catx(', ',UNSAPERF, 'BIONON');
if _UV and FACTLEU=1 then UNSAPERF = catx(', ',UNSAPERF, 'FACTLEU');
if _UV and FACTLYM=1 then UNSAPERF = catx(', ',UNSAPERF, 'FACTLYM');
if _UV and CLLSLL=1 then UNSAPERF = catx(', ',UNSAPERF, 'CLLSLL');
if _UV and ICE=1 then UNSAPERF = catx(', ',UNSAPERF, 'ICE');
if _UV and LBIMMU=1 then UNSAPERF = catx(', ',UNSAPERF, 'LBIMMU');
if _UV and NL=1 then UNSAPERF = catx(', ',UNSAPERF, 'NL');
if _UV and INL=1 then UNSAPERF = catx(', ',UNSAPERF, 'INL');
if _UV and TL1=1 then UNSAPERF = catx(', ',UNSAPERF, 'TL1');
if _UV and ITL1=1 then UNSAPERF = catx(', ',UNSAPERF, 'ITL1');
if _UV and TL2=1 then UNSAPERF = catx(', ',UNSAPERF, 'TL2');
if _UV and ITL2=1 then UNSAPERF = catx(', ',UNSAPERF, 'ITL2');
if _UV and LBBC=1 then UNSAPERF = catx(', ',UNSAPERF, 'LBBC');
if _UV and CHEM=1 then UNSAPERF = catx(', ',UNSAPERF, 'CHEM');
if _UV and COAG=1 then UNSAPERF = catx(', ',UNSAPERF, 'COAG');
if _UV and LBCRS=1 then UNSAPERF = catx(', ',UNSAPERF, 'LBCRS');
```

Step 3.2: Concatenate the Variable Names for QNAM = "UNSAPERF" in Step 9

_BLOCK A

In the UV (Unscheduled Subject Visit) dataset, concatenate these raw dataset variables into a single character variable UNSAPERF, which will later be mapped into a supplemental qualifier (_BLOCK C).

```
    if _UV and HEM=1 then UNSAPERF = catx(', ',UNSAPERF, 'HEM');
    if _UV and LBLYM=1 then UNSAPERF = catx(', ',UNSAPERF, 'LBLYM');
    if _UV and PG=1 then UNSAPERF = catx(', ',UNSAPERF, 'PG');
    if _UV and SERO=1 then UNSAPERF = catx(', ',UNSAPERF, 'SERO');
    if _UV and LS=1 then UNSAPERF = catx(', ',UNSAPERF, 'LS');
    if _UV and LP=1 then UNSAPERF = catx(', ',UNSAPERF, 'LP');
    if _UV and MRD=1 then UNSAPERF = catx(', ',UNSAPERF, 'MRD');
    if _UV and MR=1 then UNSAPERF = catx(', ',UNSAPERF, 'MR');
    if _UV and ORG=1 then UNSAPERF = catx(', ',UNSAPERF, 'ORG');
    if _UV and PET1=1 then UNSAPERF = catx(', ',UNSAPERF, 'PET1');
    if _UV and PET2=1 then UNSAPERF = catx(', ',UNSAPERF, 'PET2');
    if _UV and PE=1 then UNSAPERF = catx(', ',UNSAPERF, 'PE');
    if _UV and IWCLL=1 then UNSAPERF = catx(', ',UNSAPERF, 'IWCLL');
    if _UV and LUGANO=1 then UNSAPERF = catx(', ',UNSAPERF, 'LUGANO');
    if _UV and SCTPOST=1 then UNSAPERF = catx(', ',UNSAPERF, 'SCTPOST');
    if _UV and TB=1 then UNSAPERF = catx(', ',UNSAPERF, 'TB');
    if _UV and VS=1 then UNSAPERF = catx(', ',UNSAPERF, 'VS');
```

_BLOCK A

```
    in_UV = _UV;
    in_SV = _SV;
    if _SV or not missing(SVDTC) then output;/** with missing date from scheduled visits **/
run;

*** Below is the SAS codes for the EDC datasets with date and time;
data try2;
    attrib &attrib.;
    length SVDTC $20.;
    set
        BIO(drop=studyid siteid rename=(BIOPOSTDAT=SVDT BIOPOSTTIM=SVTM))
        BIO(drop=studyid siteid rename=(BIOPREDAT=SVDT BIOPRETIM=SVTM))
        BIONON(drop=studyid siteid rename=(NONBIODAT=SVDT NONBIOTIM=SVTM))
        CY(drop=studyid siteid rename=(CYKDAT=SVDT CYKTIM=SVTM))
        EG(drop=studyid siteid rename=(EGDAT=SVDT EGTIM=SVTM))
        EX1(drop=studyid siteid rename=(EX1ENDAT=SVDT EX1ENTIM=SVTM))
        EX1(drop=studyid siteid rename=(EX1STDAT=SVDT EX1STTIM=SVTM))
        EX2(drop=studyid siteid rename=(EX2ENDAT=SVDT EX2ENTIM=SVTM))
        EX2(drop=studyid siteid rename=(EX2STDAT=SVDT EX2STTIM=SVTM))
        EX3(drop=studyid siteid rename=(EX3ENDAT=SVDT EX3ENTIM=SVTM))
        EX3(drop=studyid siteid rename=(EX3STDAT=SVDT EX3STTIM=SVTM))
        PK(drop=studyid siteid rename=(PKDAT=SVDT PKTIM=SVTM))
        PKINF(drop=studyid siteid rename=(PKPOSTDAT=SVDT PKPOSTIM=SVTM))
        PKINF(drop=studyid siteid rename=(PKPREDAT=SVDT PKPRETIM=SVTM));
        if folder in ('VI', 'DA') then delete;/* VI=Visit Independent, DA=Disease Assessment */
        if not missing(SVDT) then SVDTC=strip(put(SVDT/24/3600,yymmdd10.));
        if not missing(SVDTC) and not missing(SVTM) then SVDTC=cats(SVDTC, 'T',
                                                        put(input(SVTM, TIME.), TOD5.));
        if not missing(SVDT) then output;
run;

*** Combine try1 and try2;
data try3;
    length SVREASOC $200.;
    set try1 try2;
       STUDYID = 'Project-Study-101';
       DOMAIN = 'SV';
       USUBJID = strip(STUDYID)||strip(substr(SUBJECT,4));
       visit = strip(put(folder, $visit.));
       visitnum = input(visit, ??visitnum.);
       visitdy = input(visit, ??visitdy.);
       if VISIT ^= 'Unscheduled' then SVPRESP = 'Y';
       if visit='Unscheduled' then unsched=1;
       if SVPRESP = 'Y' and not missing(SVDTC) then SVOCCUR = 'Y';
       if SVPRESP = 'Y' and missing(SVDTC) then SVOCCUR = 'N';
       if not missing(VISREASOC_STD) then SVREASOC=strip(VISREASOC_STD);
run;

/*read in external data, and combine with try3, eg., central tumor data, etc.*/
```

Step 2: Identify Raw Datasets With Both Dates and Times Along With Clinical Visits

Step 3: Stack the Raw Datasets and Standardize Date/Time Variable Names Into a Common Name (SVDTC)

From Block 1 of _SV.SAS

Step 4: Combine Two Data Blocks to Derive VISIT and VISITNUM for Scheduled Visits and Derive SV Variables Describing Visit-Related Information: SVPRESP, SVOCCUR, SVREASOC

Step 4.5: Import External Datasets and Combine Them with the Above Two Data Blocks

```
*** merge with DM and remove dates before informed consent date or first screening date from SV ***;
proc sort data=try3;by USUBJID SVDTC;run;
data sv0;
    set try3(where=(in_SV=1));
    by USUBJID SVDTC;
    if first.USUBJID;
run;
data try4 prescreen;
    merge try3(in=in1) sv0(keep=USUBJID SVDTC rename=(SVDTC=SVDTC1)) dm(keep=USUBJID RFICDTC in=in2);
    by USUBJID;
    if in1 and in2 and ((RFICDTC <= SVDTC) or (SVDTC1 <= SVDTC) or in_SV) then output try4;
    else if in1 and in2 then output prescreen;
run;
```

_BLOCK B

```
*** separate into scheduled, unscheduled, and missed visits;
data try_sched
     try_unsched
     try_missed;
     set try4;
     if missing(SVDTC) then output try_missed;
     else if UNSCHED then output try_unsched;
     else output try_sched;
run;
*** process scheduled visits ***;
proc sort data=try_sched;by USUBJID VISITNUM VISIT SVDTC;run;
data try_sched2;
     length STARTDTC $20.;
     retain STARTDTC;
     set try_sched;
     by USUBJID VISITNUM VISIT SVDTC;
     if first.VISIT then STARTDTC = SVDTC;
     if last.VISIT then do;
        SVSTDTC = STARTDTC;
        SVENDTC = SVDTC;
        output;
     end;
run;
proc sort data=try_sched2;by USUBJID SVSTDTC UNSCHED SVENDTC;run;
*** process unscheduled visits ***;
proc sort data=try_unsched;by USUBJID SVDTC descending SVUPDES descending UNSAPERF descending UVREAS_O;run;
data try_unsched2;
     set try_unsched;
     by USUBJID SVDTC descending SVUPDES descending UNSAPERF descending UVREAS_O;
     SVSTDTC = SVDTC;
     SVENDTC = SVDTC;
     call missing(VISITNUM);
     if first.SVDTC then output;
run;
proc sort data=try_unsched2;by USUBJID SVSTDTC UNSCHED SVENDTC;run;
*** combine and calculate VISITNUM for unscheduled visits ***;
data try5;
     retain VISITNUM1;
     set try_sched2 try_unsched2;
     by USUBJID SVSTDTC UNSCHED SVENDTC;
     VISITNUM_OLD = VISITNUM;
     VISIT_OLD = VISIT;
     if first.USUBJID then VISITNUM1 = max(0.01, VISITNUM);
     else if not missing(VISITNUM) then VISITNUM1 = VISITNUM;
run;
proc sort data=try5;by USUBJID VISITNUM1 SVSTDTC UNSCHED SVENDTC;run;
data try6;
     retain VISITNUM2;
     set try5;
     by USUBJID VISITNUM1 SVSTDTC UNSCHED SVENDTC;
     if first.VISITNUM1 then VISITNUM2 = 0;
     if UNSCHED = 1 then VISITNUM2 + 0.01;
     if VISITNUM2 > 0 then do;
       VISITNUM = VISITNUM1 + VISITNUM2;
       VISIT = catx(" ", VISIT, put(VISITNUM, 8.2));
     end;
run;
*** combine with missed visits ***;
data try7; set try6 try_missed; run;
proc sort data=try7;by USUBJID VISITNUM SVSTDTC;run;
** bring in NV(Next Visit) to add NVYN_STD(Subject Be Advancing to the Next Visit) to be stored in SUPPSV;
data try0;
     set NV(drop=studyid siteid);
         STUDYID = 'Project-Study-101';
         DOMAIN = 'SV';
         USUBJID = strip(STUDYID)||strip(substr(SUBJECT,4));
         visit = strip(put(folder, $visit.));
         visitnum = input(visit, ??visitnum.);
         visitdy = input(visit, ??visitdy.);
run;
*** merge in variables from raw SV, NV *;
proc sort data=try0;by USUBJID VISITNUM;run;
proc sort data=try4;by USUBJID VISITNUM;run;
data try;
     merge try7
           try0(keep=USUBJID VISITNUM NVYN_STD in=in1)
           try4(where=(in_SV=1) keep=USUBJID VISITNUM VISDAT SVCNTMOD SVEPCHGI in_SV in=in2);
     by USUBJID VISITNUM;
run;
*********  Programming Note: SDTM Variable: SVSTDY Needs the Derivation by the Macro Call: %get_dy;
```

Step 5: Separate the Combined Data into Two Data Blocks: One With Scheduled Visits Only, Another With Unscheduled Visits Only

Step 6: For Each Scheduled Visit, Derive SVSTDTC From the Earliest Date/Time and Derive SVENDTC From the Latest Date/Time

Step 6: For Each Unscheduled Visit, Set Both SVSTDTC and SVENDTC Equal to SVDTC

Step 7: Combine the Scheduled Visits and Unscheduled Visits Blocks and Derive VISIT and VISITNUM for Unscheduled Visits

_BLOCK B

Step 8: Derive SVSTDY and SVENDY

From Block 3 of _SV.SAS

```
%get_dy(_DATEVAR=SVSTDTC,_DAYVAR=SVSTDY);
*********  Programming Note: SDTM Variable: SVENDY Needs the Derivation by the Macro Call: %get_dy;
%get_dy(_DATEVAR=SVENDTC,_DAYVAR=SVENDY);
proc sort data=try;by studyid usubjid VISITNUM;run;
*** Output permanent datasets to: wsdtm ***;
data wsdtm.&domain.(keep=&keep label=&label);
     attrib &attrib.;
     set try;
     format _all_;
     informat _all_;
run;
```

Step 10: Output Permanent Dataset: SV

From Block 4 of _SV.SAS

```
data supp&domain.;
     attrib &attrib_supp.;
     set try;
     rdomain = "&domain.";
     idvar = 'VISITNUM';
     idvarval = strip(put(VISITNUM, best.));
     if not missing(NVYN_STD) then do;
          qnam='NXTVISYN';
          qlabel='Subject Advancing to Next Visit?';
          qval=strip(NVYN_STD);
          qorig='CRF';
          qeval='';
          output;
     end;
     if not missing(VISDAT) then do;
          qnam='VISDTC';
          qlabel='Visit Date';
          qval=strip(put(VISDAT/24/3600,yymmdd10.));
          qorig='CRF';
          qeval='';
          output;
     end;
     if not missing(UVREAS_O) then do;
          qnam='UNSREASO';
          qlabel='Unscheduled Visit: Other Reason, Specify';
          qval=strip(UVREAS_O);
          qorig='CRF';
          qeval='';
          output;
     end;
     if not missing(VISREASOC_AE) then do;
          qnam='VISMAEID';
          qlabel='Missed Visit: Adverse Event, Specify';
          qval=strip(VISREASOC_AE);
          qorig='CRF';
          qeval='';
          output;
     end;
```

Step 9: Map Other Raw Dataset Variables into SUPPSV

```
     if not missing(UNSAPERF) then do;
          qnam='UNSAPERF';
          qlabel='Unscheduled Assessments Performed';
          qval=strip(UNSAPERF);
          qorig='CRF';
          qeval='';
          output;
     end;
```

_BLOCK C

Variable UNSAPERF was derived in _BLOCK A. Here it is output as QVAL for QNAM = "UNSAPERF" for SUPPSV.

```
     if not missing(VISREASOC_O) then do;
          qnam='VISMOTHS';
          qlabel='Missed Visit: Other Reason, Specify';
          qval=strip(VISREASOC_O);
          qorig='CRF';
          qeval='';
          output;
     end;
     if not missing(VISCNTMD_O) then do;
          qnam='VISTYPOS';
          qlabel='Type of Visit: Other, Specify';
          qval=strip(VISCNTMD_O);
          qorig='CRF';
          qeval='';
          output;
     end;
run;
```

From Block 5 of _SV.SAS

```
proc sort data=&suppdomain.;by usubjid idvarval qnam;run;
data wsdtm.&suppdomain.(keep=&keep_supp label=&label_supp);
       attrib &attrib_supp.;
       set &suppdomain.;
     format _all_;
     informat _all_;
run;
```

Step 10: Output Permanent Dataset: SUPPSV